


FACULTY OF TECHNOLOGY AND ENGINEERING

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

A.Y. 2023-24 [ODD]

LAB MANUAL

CE261 DATA STRUCTURE & ALGORITHMS

Semester: III **Academic Year: 2023**

Subject Code: CE261 **Subject Name: Data Structures & Algorithms**

Student Id:22DCE006 **Student Name: Probin T Bhagchandani**

PRACTICAL INDEX

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
1.	Implement Linear Search and Binary Search using array data structure. **Supplementary Experiment: [L: A]** 1. https://www.codechef.com/problems/SEGM01 2. https://www.hackerrank.com/contests/launchpad-1-winter-challenge/challenges/binary-search-advanced 3. https://www.codechef.com/problems/CHEFHCK2 4. https://www.codechef.com/problems/SNAKEEAT 5. https://www.codechef.com/problems/DIVSET					
2.	For a given array Bl, B2, ⋯, BM of length at least 3, let's define its weight as the largest value of (Bi−Bj)·(Bj−Bk) over all possible triples (i, j, k) with l ≤ i, j, k ≤ M and i ≠ j, j ≠ k, k ≠ i. You are given a sorted array Al, A2, ⋯,AN (that is, Al ≤ A2 ≤⋯≤ AN). Calculate the sum of weights of all contiguous subarrays of A of length at least 3. That is, count the sum of weights of arrays [Ai, Ai+l, ⋯, Aj] over all l					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	$\leq i < j \leq N$ with $j-i \geq 2$. **Input:** The first line of input contains a single integer T denoting the number of test cases. The description of T test cases follows. The first line of each test case contains an integer N. The second line of each test case contains N space-separated integers A1, A2, …, AN. **Output:** For each test case, print a single line containing the sum of weights of all subarrays of A of length at least 3. **Constraints** • $1 \leq T \leq 1000$ • $3 \leq N \leq 3000$ • $1 \leq A1 \leq A2 \cdots \leq AN \leq 106$. • Sum of N over all test cases won't exceed 6000 **Sample Input 1** 2 4 1 2 3 4 5 1 42 69 228 2021 **Sample Output 1** 4 1041808					
3.	3.1 Implement following operations of singly linked list. (a) Insert a node at front (b) Insert a node at end (c) Insert a node after given node information					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	(d) Delete a node at front (e) Delete a node at last 3.2 Implement following operations of doubly linked list. (a) Insert a node at front (b) Insert a node at end (c) Insert a node after given node information (d) Delete a node at front (e) Count number of nodes 3.3 Implement following operations of circular singly linked list. (a) Inserting a node at front (b) Delete a node at end **Note: Display content of linked list after each operation.**					
4.	Implement Sorting Algorithm(s). (a) Bubble Sort (b) Selection Sort (c) Quick Sort (d) Merge Sort **Supplementary Experiment:** **1.** https://www.codechef.com/problems/TSORT **[L: M]** **2.** https://www.codechef.com/problems/MRGSRT **[L: A]**					
5.	Chef and his little brother are playing with sticks. They have total N sticks. Length of i-th stick is Ai. Chef asks his brother to choose any four sticks and to make a rectangle with those sticks its sides. Chef warns his brother to not to break any of the sticks, he has to use sticks as a whole. Also, he wants that the rectangle formed should have the maximum possible area among all the rectangles that Chef's brother can make. Chef's little brother takes this challenge up and overcomes it. Can you also do so? That is, you have to					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	tell whether it is even possible to create a rectangle? If yes, then you have to tell the maximum possible area of rectangle. Input ● The first line contains a single integer T denoting the number of test-cases. T test cases follow. ● The first line of each test case contains a single integer N denoting the number of sticks. ● The second line of each test case contains N space-separated integers A1, A2, ..., AN denoting the lengths of sticks. Output ● For each test case, output a single line containing an integer representing the maximum possible area for rectangle or -1 if it's impossible to form any rectangle using the available sticks. Input 2 5 1 2 3 1 2 4 1 2 2 3 Output 2 -1					
6.	1. Implement basic operations (push (), pop () and display ()) of stack using array. 2. Implement basic operations					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	(push (), pop () and display ()) of stack using linked list. **Supplementary Experiment:** 1. https://www.codechef.com/problems/SUDBOOKS **[L: A]**					
7.	Chef has a string which contains only the characters '{', '}', '[', ']', '(' and ')'. Now Chef wants to know if the given string is balanced or not. If is balanced then print 1, otherwise print 0. A balanced parenthesis string is defined as follows: ● The empty string is balanced ● If P is balanced then (P), {P}, [P] is also balanced ● if P and Q are balanced PQ is also balanced For example "([])", "({})[()]" are balanced parenthesis strings while "([{]})", "())" are not balanced. Input The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows. The first and only line of each test case contains a single string Output For each test case, print a single line containing the answer. Input: 4 () ([)] ([{}()])[{}] [{{}] Output: 1 0					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	1 0					
8.	1. Implement basic operations (enqueue (), dequeue () and display ()) of queue using array. 2. Implement basic operations (enqueue (), dequeue () and display ()) of queue using linked list. 3. Implement basic operations (enqueue (), dequeue () and display ()) of circular queue using array. **Supplementary Experiment:** 1. https://www.codechef.com/problems/CHFQUEUE **[L: A]**					
9.	There are *N* people, numbered from 11 to *N*. They go to a cinema hall. Each of them buys a ticket, which has a number written on it. The number on the ticket of the i^{th} person is A_i. There are infinite seats in the cinema hall. The seats are numbered sequentially starting from 11. All the *N* people stand in a queue to get their respective seats. Person 11 stands at the front of the queue, Person 22 stands in the second position of the queue, so on up to Person *N* who stands at the rear of the queue. They were given seats in this manner: Let the number on the ticket of the person currently standing in front of the queue be *X*. If the X^{th} seat is empty, the person gets out of the queue and takes the X^{th} seat. Otherwise, the person goes to the rear of the queue, and the number on his ticket is incremented by one - that is, it becomes *X*+1.					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	Print the seat number occupied by each of the *N* people. Input ● The first line of input contains a single integer T denoting the number of test cases. The description of T test cases follows. ● The first line of each test case contains a single integer N. ● The second line of each testcase line contains N space-separated integers A_1, A_2, …, A_N. Output ● For each test case, print a single line containing *N* space-separated integers, where the i^{th} integer denotes the seat number finally occupied by the Person *i*. Input 4 5 1 2 3 2 4 4 4 1 3 2 3 1 1 1 5 2 5 1 5 2 Output 1 2 3 5 4 4 1 3 2 1 2 3 2 5 1 6 3					
10.	Implement Binary Search Tree (BST) using following operations. (a) Insert					

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
	(b) Search (c) Traversal (Inorder, Preorder, Postorder)					
11.	Implement a Graph to perform following operations. 1. Adjacency list representation 2. Apply DFS and BFS on the given graph.					
12.	In an array of 20 elements, arrange 15 different values, which are generated randomly between 1,00,000 to 9,99,999. Use hash function to generate key and linear probing to avoid collision. H(x) = (x mod 18) + 2. Write a program to input and display the final values of array.					

PRACTICAL-1

AIM: Implement Linear Search and Binary Search using array data structure.

PROGRAM CODE:

For Linear Search

```
import java.util.*;
public class first
{
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("HELLO WORLD");
      int i,j,k;
      int a[]={20,30,50,70,90};
      int x=70;
      int len=a.length;
      System.out.println("Length : "+len);
      for(i=0 ; i<len ; i++)
      {
         if(a[i]==x)
         {
            System.out.println("Element is at "+i+" location");
         }
         else
         {
            System.out.println("Element not found at "+i);
         }
      }
    }
}
```

For Binary Search

```
import java.util.*;
public class first
{
   public static void main(String args[])
   {
      Scanner sc=new Scanner(System.in);
      System.out.println("HELLO WORLD");
      int i,j,k;
```

```
int a[]={30,20,70,50,90};
int x=20;
int len=a.length;
System.out.println("Length : "+len);
System.out.println("Sorted array");
Arrays.sort(a);
for(i=0 ; i<len ; i++)
{
   System.out.println("Element is at "+a[i]);
}
int ub=a.length-1 , lb=0 , mid;
mid =(int)(ub+lb) /2 ;
int  key=a[mid];

System.out.println("Middle Element is "+key);
for(i=0 ; i<a.length; i++)
{
   if(a[mid]==x)
   {
      System.out.println("Element is at "+i);
      break;
   }
   else if(x<a[mid])
   {
      ub=mid-1 ;
      mid =(int)(ub+lb) /2 ;
      break;
   }
   else
   {
      ub=mid+1 ;
      mid =(int)(ub+lb) /2 ;
      break;
   }
}
for(i=0 ; i<len ; i++)
{
   if(a[i]==x)
   {
      System.out.println("Element is at "+i+" location");
   }
   else
   {
      System.out.println("Element not found at "+i);
   }
}
```

```
    }
}
```

OUTPUT:

For Linary Search

```
HELLO WORLD
Length : 5
Element not found at 0
Element not found at 1
Element not found at 2
Element is at 3 location
Element not found at 4
PS D:\java>
```

For Binear Search

```
PS D:\java> javac first.java
PS D:\java> java first
HELLO WORLD
Length : 5
Sorted array
Element is at 20
Element is at 30
Element is at 50
Element is at 70
Element is at 90
Middle Element is 50
Element is at 0 location
Element not found at 1
Element not found at 2
Element not found at 3
Element not found at 4
PS D:\java>
```

CONCLUSION: From this practical, we learned about the concepts of linear search and its working. We also learned the concept of Binary search. The time complexity of binary search is lesser and is faster.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-2

AIM: For a given array B1, B2, ..., BM of length at least 3, let's define its weight as the largest value of (Bi−Bj)·(Bj−Bk) over all possible triples (i, j, k) with 1 ≤ i, j, k ≤ M and i ≠ j, j ≠ k, k ≠ i. You are given a sorted array A1, A2, ...,AN (that is, A1 ≤ A2 ≤...≤ AN). Calculate the sum of weights of all contiguous subarrays of A of length at least 3. That is, count the sum of weights of arrays [Ai, Ai+1, ..., Aj] over all 1 ≤ i < j ≤ N with j−i ≥ 2.

PROGRAM CODE:

```
import java.util.*;
public class first
{
   public static void main(String args[])
   {
      System.out.println("22DCE006");
      int a[]={1,2,3,4,5};
      int n=a.length;
      int sum=0;
      System.out.print("\n");
      int b[]=new int[3];
      System.out.println();
      System.out.println("Sub-Array is: ");
      int i,j,k;
      for(i=0 ; i<n-2 ; i++)
      {
         for(j=i+2 ; j<n ; j++ )
         {
            for(k=i ; k<=j ; k++)
            {
               System.out.print(" "+a[k]);

            }
            System.out.println();
         }
         System.out.println();
      }

      for(i=0;i<n-2;i++)
```

```
    {
       for(k=i+2;k<n;k++)
       {
         int w=0,m=0;
          for(j=i+1 ; j<k ; j++)
          {
             w = (a[i]-a[j])*(a[j]-a[k]);
             if(w > m)
             {
                m = w;
             }
          }
          sum=sum+m;
       }
    }
    System.out.println("Weight's Sum is: "+sum);
  }
}
```

OUTPUT:

```
PS C:\java> javac first.java
PS C:\java> java first
22DCE006


Sub-Array is:
 1 2 3
 1 2 3 4
 1 2 3 4 5

 2 3 4
 2 3 4 5

 3 4 5

Weight's Sum is: 11
PS C:\java>
```

TEST SUITES

2
4
1 2 3 4
5
1 42 69 228 2021

CONCLUSION:

From this practical we can conclude that Binary Search is useful for dividing an array into contagious sub-arrays and their comparison.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-3

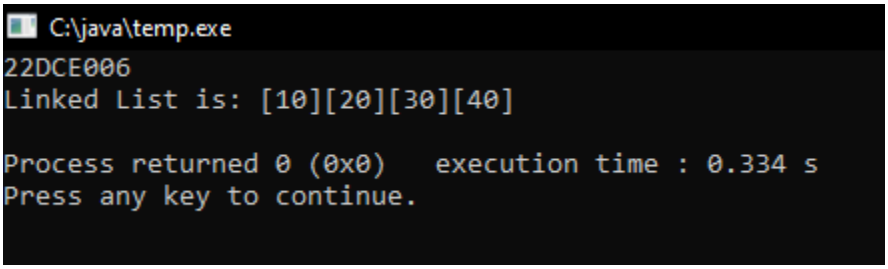
AIM: 3.1 Implement following operations of singly linked list.

(a) Insert a node at front

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=head;
   head=p;
}

void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}
int main()
{
   cout<<"22DCE006";
   push(40);
   push(30);
   push(20);
   push(10);
   cout<<"\n";
   print();
   cout<<"\n";
}
```

OUTPUT:



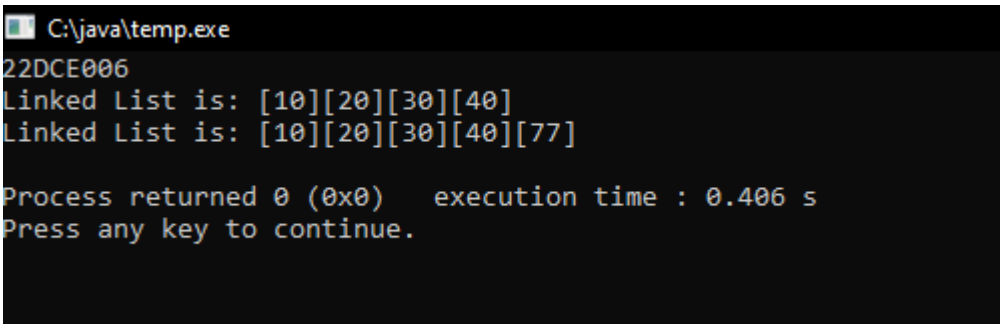
(b) Insert a node at end

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=head;
   head=p;
}

void insert_atlast(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    struct Node *temp=head;
    while(temp->next!=NULL)
    {
       temp=temp->next;
    }
    temp->next=p;
    p->next=NULL;
}
void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
```

```
    {
        cout<<"["<<temp->data<<"]";
        temp=temp->next;
    }
}

int main()
{
    cout<<"22DCE006";
    cout<<"\n";
    push(40);
    push(30);
    push(20);
    push(10);
    print();
    cout<<"\n";
    insert_atlast(77);
    print();
    cout<<"\n";
}
```

OUTPUT:



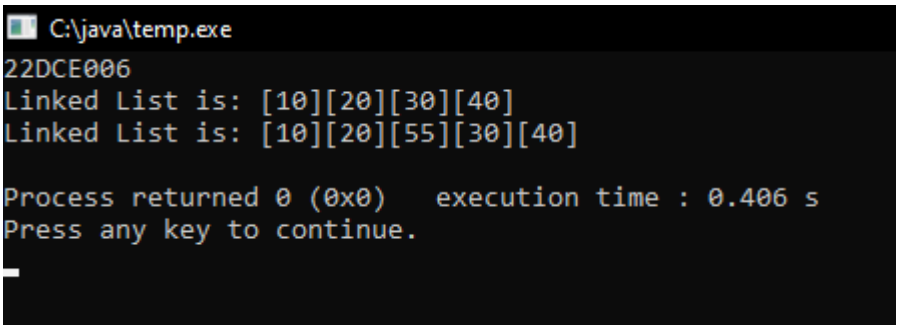
(c) Insert a node after given node information

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
    int data;
    Node *next;
};
struct Node *head=NULL;
void push(int data)
{
    struct Node *p=(struct Node*)malloc(sizeof(struct Node));
```

```
   p->data=data;
   p->next=head;
   head=p;
}


void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}


void insert_between(struct Node *temp,int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=temp->next;
   temp->next=p;
}

int main()
{
   cout<<"22DCE006";
   cout<<"\n";
   push(40);
   push(30);
   push(20);
   push(10);
   print();
   cout<<"\n";
   insert_between(head->next,55);
   print();
   cout<<"\n";
}
```

OUTPUT:



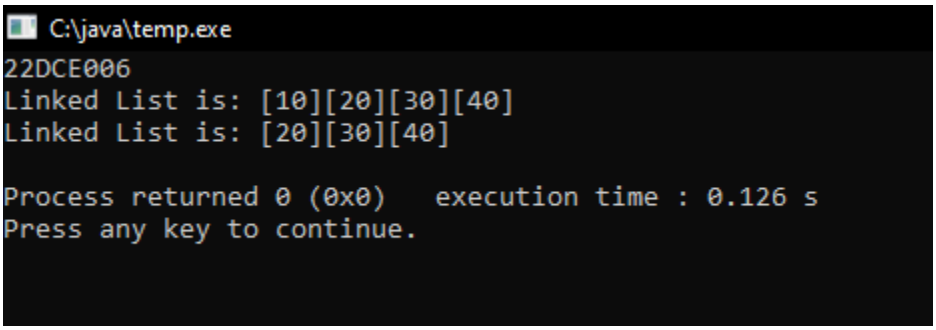
(d) Delete a node at front

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=head;
   head=p;
}

void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}
void deletefun_atfront()
{
   head=head->next;
}
int main()
{
```

```
    cout<<"22DCE006";
    cout<<"\n";
    push(40);
    push(30);
    push(20);
    push(10);
    print();
    cout<<"\n";
    deletefun_atfront();
    print();
    cout<<"\n";
}
```

OUTPUT:



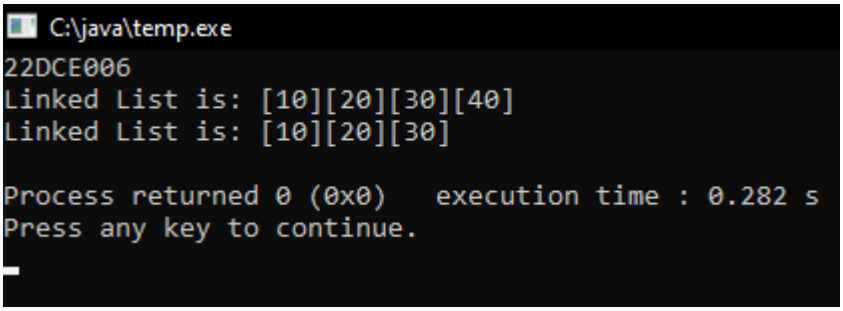
(e) Delete a node at last

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
    int data;
    Node *next;
};
struct Node *head=NULL;
void push(int data)
{
    struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    p->next=head;
    head=p;
}
```

```
void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}


void delete_atlast()
{
   struct Node *temp=head;
   while(temp->next->next!=NULL)
   {
      temp=temp->next;
   }
   temp->next=NULL;
}
int main()
{
   cout<<"22DCE006";
   cout<<"\n";
   push(40);
   push(30);
   push(20);
   push(10);
   print();
   cout<<"\n";
   delete_atlast();
   print();
   cout<<"\n";
}
```

OUTPUT:



3.2 Implement following operations of doubly linked list.

(a) Insert a node at front

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
   Node *prev;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=NULL;
   p->prev=NULL;
   if(head==NULL)
   {
      head=p;
   }
   else
   {
      p->next=head;
      head->prev=p;
      head=p;
   }
}

void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}
int main()
{
   cout<<"22DCE006";
   push(40);
```

```
    push(30);
    push(20);
    push(10);
    cout<<"\n";
    print();
    cout<<"\n";
}
```

OUTPUT:

```
C:\java\one.exe
22DCE006
Linked List is: [10][20][30][40]

Process returned 0 (0x0)   execution time : 0.125 s
Press any key to continue.
```

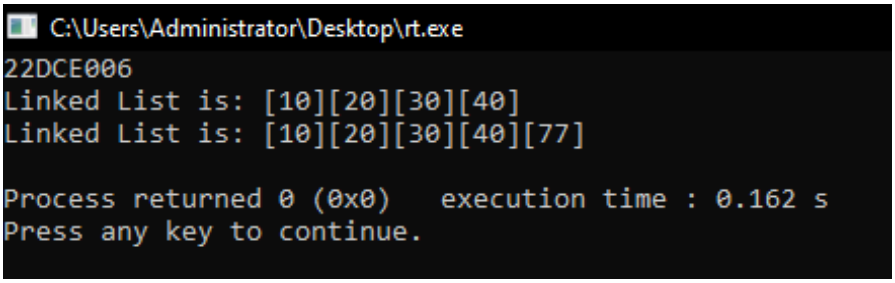
(b) Insert a node at end

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
    int data;
    Node *next;
    Node *prev;
};
struct Node *head=NULL;
void push(int data)
{
    struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    p->next=NULL;
    p->prev=NULL;
    if(head==NULL)
    {
        head=p;
    }
    else
    {
        p->next=head;
        head->prev=p;
        head=p;
    }
```

```
}

void insert_atlast(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    struct Node *temp=head;
    while(temp->next!=NULL)
    {
       temp=temp->next;
    }
    temp->next=p;
    p->next=NULL;
}
void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}
int main()
{
   cout<<"22DCE006";
   cout<<"\n";
   push(40);
   push(30);
   push(20);
   push(10);
   print();
   cout<<"\n";
   insert_atlast(77);
   print();
   cout<<"\n";
}
```

OUTPUT:



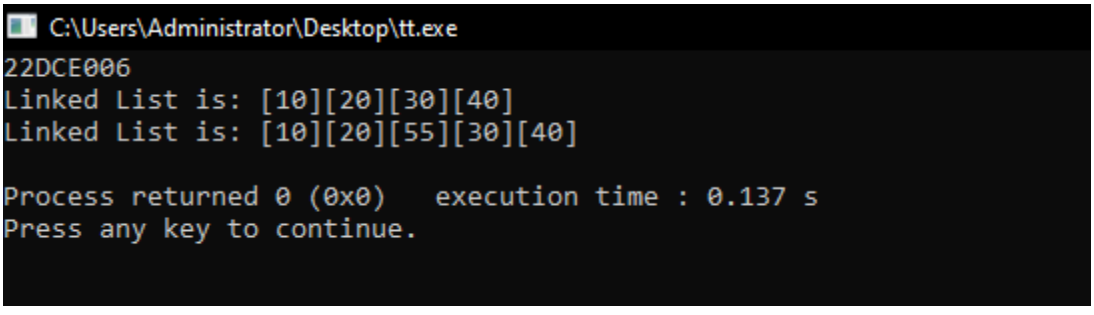
(c) Insert a node after given node information

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
   Node *prev;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=NULL;
   p->prev=NULL;
   if(head==NULL)
   {
      head=p;
   }
   else
   {
      p->next=head;
      head->prev=p;
      head=p;
   }
}
void insert_between(struct Node *temp,int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=NULL;
   p->prev=NULL;
   p->next=temp->next;
```

```
    temp->next=p;
    p->prev=temp;

}
void print()
{
    cout<<"Linked List is: ";
    struct Node *temp=head;
    while(temp!=NULL)
    {
        cout<<"["<<temp->data<<"]";
        temp=temp->next;
    }
}

int main()
{
    cout<<"22DCE006";
    cout<<"\n";
    push(40);
    push(30);
    push(20);
    push(10);
    print();
    cout<<"\n";
    insert_between(head->next,55);
    print();
    cout<<"\n";
}
```

OUTPUT:



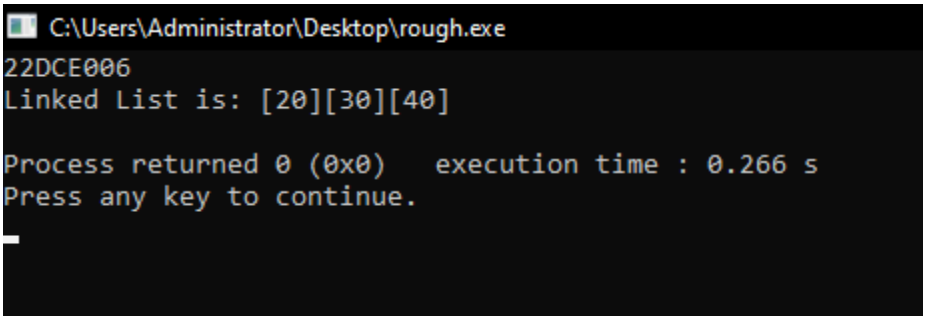
(d) Delete a node at front

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
   Node *prev;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=NULL;
   p->prev=NULL;
   if(head==NULL)
   {
      head=p;
   }
   else
   {
      p->next=head;
      head->prev=p;
      head=p;
   }
}
void deletefun_atfront()
{
   head=head->next;
   head->prev=NULL;
}
void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
}
int main()
{
```

```
    cout<<"22DCE006";
    push(40);
    push(30);
    push(20);
    push(10);
    cout<<"\n";
    deletefun_atfront();
    print();
    cout<<"\n";
}
```

OUTPUT:



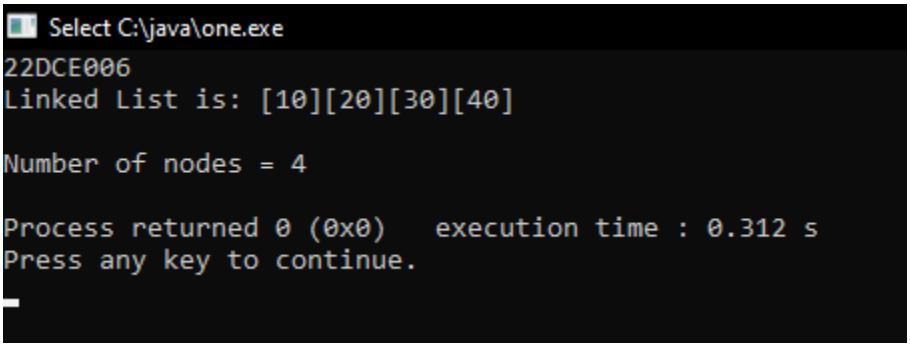
(e) Count number of nodes

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
    int data;
    Node *next;
    Node *prev;
};
struct Node *head=NULL;
void push(int data)
{
    struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    p->next=NULL;
    p->prev=NULL;
    if(head==NULL)
    {
        head=p;
    }
    else
    {
        p->next=head;
        head->prev=p;
```

```
        head=p;
    }
}

void print()
{
    int counter=0;
    cout<<"Linked List is: ";
    struct Node *temp=head;
    while(temp!=NULL)
    {
        cout<<"["<<temp->data<<"]";
        temp=temp->next;
        counter++ ;
    }
    cout<<"\n\nNumber of nodes = "<<counter;
}
int main()
{
    cout<<"22DCE006";
    push(40);
    push(30);
    push(20);
    push(10);
    cout<<"\n";
    print();
    cout<<"\n";
}
```

OUTPUT:



3.3 Implement following operations of circular singly linked list.

(a) Inserting a node at front

CODE:

```
#include<iostream>
using namespace std;
struct Node
{
   int data;
   Node *next;
};
struct Node *head=NULL;
void push(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=head;
   head=p;
}

void insert_atlast(int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
    p->data=data;
    struct Node *temp=head;
    while(temp->next!=NULL)
    {
       temp=temp->next;
    }
    temp->next=p;
    p->next=NULL;
}
void print()
{
   cout<<"Linked List is: ";
   struct Node *temp=head;
   while(temp!=NULL)
   {
      cout<<"["<<temp->data<<"]";
      temp=temp->next;
   }
   temp=head;
   cout<<"["<<temp->data<<"]";
}
```

```
void insert_between(struct Node *temp,int data)
{
   struct Node *p=(struct Node*)malloc(sizeof(struct Node));
   p->data=data;
   p->next=temp->next;
   temp->next=p;
}

int main()
{
   cout<<"22DCE006";
   cout<<"\n";
   push(50);
   push(40);
   push(30);
   push(20);
   push(10);
   print();
   cout<<"\n";
}
```

OUTPUT:

```
22DCE006
Linked List is: [10][20][30][40][50][10]

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

(b) Deleting a node at end

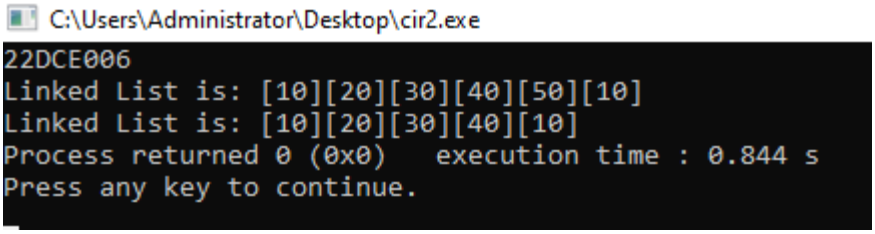
CODE:

```
#include<iostream>
using namespace std;
struct Node
{
int data;
Node *next;
};
struct Node *head=NULL;

void push(int data)
```

```
{
struct Node *p=(struct Node*)malloc(sizeof(struct Node));
p->data=data;
p->next=head;
head=p;
}

void delete_atlast()
{
   struct Node *temp = head;
   while(temp->next->next!=NULL)
   {
      temp=temp->next;
   }
   temp->next=NULL;
}
void print()
{
cout<<"Linked List is: ";
struct Node *temp=head;
while(temp!=NULL)
{
cout<<"["<<temp->data<<"]";
temp=temp->next;
}
temp=head;
cout<<"["<<temp->data<<"]";
}

int main()
{
cout<<"22DCE006";
cout<<"\n";
push(50);
push(40);
push(30);
push(20);
push(10);
print();
cout<<"\n";
```

```
delete_atlast();
print();
}
```

OUTPUT:



Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-4

AIM: 3.1 Implement Sorting Algorithm(s).

(a) Bubble Sort
(b) Selection Sort
(c) Quick Sort
(d) Merge Sort

Code:

```
public class codingprac
{
   public static void BubbleSort(int a[])
   {
      for(int i=0 ; i<a.length-1 ; i++)
      {
         for(int j=0 ; j<a.length-1-i ; j++)
         {
            if(a[j]>a[j+1])
            {
               int temp=a[j];
               a[j]=a[j+1];
               a[j+1]=temp;
            }
         }
      }
   }

   public static void SelectionSort(int a[])
   {
      for(int i=0 ; i<a.length ; i++)
      {
         int min=i;
         for(int j=i+1 ; j<a.length ; j++)
         {
            if(a[j]<a[min])
            {
               min=j;
            }
```

```
        }
        int temp;
        temp=a[min];
        a[min]=a[i];
        a[i]=temp;
    }
}

public static void InsertionSort(int a[])
{
    for(int i=1 ; i<a.length ; i++)
    {
        int temp=a[i];
        int j=i-1;
        while((j>=0) && (temp < a[j]))
        {
            a[j+1]=a[j];
            j--;
        }
        a[j+1]=temp;
    }
}

public static void MergeSort(int a[] , int starting , int ending)
{
    if(starting>=ending)
    {
        return;
    }
    int mid=starting + (ending-starting)/2 ;

    MergeSort(a, starting, mid);
    MergeSort(a, mid+1, ending);
    merge(a, starting, mid, ending);
}

public static  void merge(int a[] , int starting , int mid , int ending)
{
    int temp[]=new int[ending-starting +1];
    int i=starting;
```

```
    int j=mid+1;
    int k=0;

    while(i<=mid && j<=ending)
    {
       if (a[i]<a[j])
       {
          temp[k]=a[i];
          i++;
          k++;
       }
       else
       {
          temp[k]=a[j];
          j++;
          k++;
       }
    }
    while(i<=mid)
    {
       temp[k++]=a[i++];
    }
    while(j<=ending)
    {
       temp[k++]=a[j++];
    }
    for(k=0 , i=starting ; k<temp.length ; k++ , i++)
    {
       a[i]=temp[k];
    }
}
public static void QuickSort(int a[] , int starting , int ending)
{
    if(starting>=ending)
    {
       return;
    }
    int pivot_index=partition(a,starting,ending);
    QuickSort(a, starting, pivot_index-1);
    QuickSort(a, pivot_index+1 , ending );
```

```
}

public static int partition(int a[] , int starting , int ending)
{
   int pivot=a[ending];
   int i=starting-1;
   for(int j=starting ; j<ending ; j++)
   {
      if(a[j]<=pivot)
      {
         i++;
         int temp=a[j];
         a[j]=a[i];
         a[i]=temp;
      }
   }
   i++;
   int temp=pivot;
   a[ending]=a[i];
   a[i]=temp;
   return i;
}

public static void display(int a[])
{
   for(int i=0 ; i<a.length ; i++)
   {
      System.out.print(" "+a[i]);
   }
}
public static void main(String args[])
{
   int a[]={5,4,1,3,2,6};
   System.out.println("\nUsing Bubble Sort");
   BubbleSort(a);
   display(a);
   System.out.println("\nUsing Selection Sort");
   SelectionSort(a);
   display(a);
   System.out.println("\nUsing Insertion Sort");
```

```
        InsertionSort(a);
        display(a);
        System.out.println("\nUsing Merge Sort");
        MergeSort(a, 0, a.length-1);
        display(a);
        System.out.println("\nUsing Quick Sort");
        QuickSort(a,0,a.length-1);
        display(a);
    }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac codingprac.java
PS D:\Probin's Work\Java Programming> java codingprac

Using Bubble Sort
 1 2 3 4 5 6
Using Selection Sort
 1 2 3 4 5 6
Using Insertion Sort
 1 2 3 4 5 6
Using Merge Sort
 1 2 3 4 5 6
Using Quick Sort
 1 2 3 4 5 6
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-5

AIM: Chef and his little brother are playing with sticks. They have total N sticks. Length of i-th stick is Ai. Chef asks his brother to choose any four sticks and to make a rectangle with those sticks its sides. Chef warns his brother to not to break any of the sticks, he has to use sticks as a whole. Also, he wants that the rectangle formed should have the maximum possible area among all the rectangles that Chef's brother can make. Chef's little brother takes this challenge up and overcomes it. Can you also do so? That is, you have to tell whether it is even possible to create a rectangle? If yes, then you have to tell the maximum possible area of rectangle.

Input

• The first line contains a single integer T denoting the number of test-cases. T test cases follow.

• The first line of each test case contains a single integer N denoting the number of sticks.

• The second line of each test case contains N space-separated integers A1, A2, ..., AN denoting the lengths of sticks.

Output

• For each test case, output a single line containing an integer representing the maximum possible area for rectangle or -1 if it's impossible to form any rectangle using the available sticks.

Input

```
2
5
1 2 3 1 2
4
1 2 2 3
```

Output

```
2
-1
```

PROGRAM CODE:

```
import java.util.*;
public class clg
{
   public static  int Max_Area(int array[] , int size)
   {
      int a=-1,b=-1;
      for (int i=0 ; i<size ; i++ )
      {
         if(array[i]==array[i+1])
         {
            if(a==-1)
            {
               a=array[i];
            }
            else if(b==-1)
            {
               b=array[i];
            }
         }
      }
      if(a!=-1 && b!=-1)
      {
         return a*b;
      }
      return -1;
   }
   public static void main(String args[])
   {
      Scanner sc = new Scanner(System.in);
      int size=5;
      int array[]=new int[size];
      for(int j = 0;j < size;++j)
      {
         System.out.print("Enter the number : ");
         array[j]=sc.nextInt();
      }
      Arrays.sort(array);
      int result = Max_Area(array, size-1);
      System.out.println("Area = "+result);
      sc.close();
   }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> javac clg.java
PS D:\Probin's Work\Java Programming> java clg
Enter the number : 1
Enter the number : 2
Enter the number : 3
Enter the number : 2
Enter the number : 1
Area = 2
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff

PRACTICAL-6

AIM:
1. Implement basic operations (push (), pop () and display ()) of stack using array.
2. Implement basic operations (push (), pop () and display ()) of stack using linked list.

PROGRAM CODE:

1)Using Array

Program Code:

```
public class example
{
    int[] array;
    int top;
   public example(int capacity) {
      array = new int[capacity];
      top = -1;
   }
   public void push(int element) {
      if (top == array.length - 1) {
         System.out.println("Stack is full. Cannot push element.");
      } else {
         top++;
         array[top] = element;
         System.out.println("Pushed element: " + element);
      }
   }
   public int pop() {
      if (top == -1) {
         System.out.println("Stack is empty. Cannot pop element.");
         return -1;
      } else {
         int poppedElement = array[top];
         top--;
         System.out.println("Popped element: " + poppedElement);
         return poppedElement;
      }
   }
   public boolean isEmpty() {
      return (top == -1);
   }
```

```
  public void display()
   {
      if(isEmpty())
      {
         System.out.println("Stack is empty");
      }
      else
      {
         System.out.print("n Stack : ");
         for(int i=0; i<=top; i++)
         {
            System.out.print(" | "+array[i]);
         }
      }
   }
  public static void main(String[] args)
   {
     example stack = new example(5);

     stack.push(10);
     stack.push(20);
     stack.push(30);
     stack.pop();
     stack.push(40);
     stack.push(50);
     stack.display();
   }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac example.java
PS D:\Probin's Work\Java Programming> java example
Pushed element: 10
Pushed element: 20
Pushed element: 30
Popped element: 30
Pushed element: 40
Pushed element: 50
n Stack :  | 10 | 20 | 40 | 50
PS D:\Probin's Work\Java Programming>
```

2)Using LinkedList

Program Code:

```
public class jp
{
   public static class Node
   {
      int data;
      Node next;
      public Node(int data)
      {
         this.data = data;
         this.next=null;
      }
   }
   public static Node head;
   public static Node tail;
   public int counter;
   public void push(int data)
   {
      Node newNode=new Node(data);
      counter++;
      if(head==null)
      {
         head=tail=newNode;
         return;
      }
      newNode.next=head;
      head=newNode;
   }
   public void pop()
   {
      if(head==null)
      {
         System.out.println("LinkedList is empty");
      }
      else if(head==tail)
      {
         System.out.println("NULL");
      }

      head=head.next;
      counter-- ;
   }
   public void display()
```

```
    {
        if(head==null)
        {
            System.out.println("Stack is empty");
            return;
        }
        Node temp=head;
        while (temp!= null)
        {
            System.out.println(temp.data+" ");
            temp=temp.next;
        }
    }
    public static void main(String args[])
    {
        jp l1=new jp();
        l1.push(40);
        l1.push(30);
        l1.push(20);
        l1.push(10);
        System.out.println("\nSTACK: ");
        l1.pop();
        l1.display();
    }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac jp.java
PS D:\Probin's Work\Java Programming> java jp

STACK:
20
30
40
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-7

AIM: Chef has a string which contains only the characters '{', '}', '[', ']', '(' and ')'. Now Chef wants to know if the given string is balanced or not. If is balanced then print 1, otherwise print 0.A balanced parenthesis string is defined as follows:
• The empty string is balanced
• If P is balanced then (P), {P}, [P] is also balanced
• if P and Q are balanced PQ is also balanced
For example "([])", "({})[()]" are balanced parenthesis strings while " ([{]})", "())" are not balanced.
Input The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows. The first and only line of each test case contains a single string
Output
For each test case, print a single line containing the answer.
Input:
4
()
([])
([{}()])[{}]
[{{}]
Output:
1
0
1
0

Program Code:

```
public class basic
{
   public static boolean check_if_Balanced(String s1)
   {
      int i = -1;
      char[] stack = new char[s1.length()];
      for (char ch : s1.toCharArray())
      {
         if (ch == '(' || ch == '{' || ch == '[')
         {
            stack[++i] = ch;
         }
         else
         {
            if (i >= 0 && ((stack[i] == '(' && ch == ')')  || (stack[i] == '{' && ch == '}') ||
            (stack[i] == '[' && ch == ']')))
```

```
            {
               i--;
            }
            else
            {
               return false;
            }
         }

      }
      return true;
   }
   public static void main(String[] args)
   {
      String s1= "{()}[]";
      if (check_if_Balanced(s1)==true)
      {
         System.out.println("Given string is Balanced");
      }
      else
      {
         System.out.println("Given string is Not Balanced");
      }
   }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac basic.java
PS D:\Probin's Work\Java Programming> java basic
Given string is Balanced
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-8

AIM:

1. Implement basic operations (enqueue (), dequeue () and display ()) of queue using array.

2. Implement basic operations (enqueue (), dequeue () and display ()) of queue using linked list.

3. Implement basic operations (enqueue (), dequeue () and display ()) of circular queue using array.

1)Queue Using Array

Program Code:

```
public class  trial
{
   int size = 5;
   int items[] = new int[size];
   int front, rear;

   trial() {
     front = -1;
     rear = -1;
   }
   boolean isFull() {
     if (front == 0 && rear == size - 1) {
       return true;
     }
     return false;
   }
   boolean isEmpty() {
     if (front == -1)
       return true;
     else
       return false;
   }

   void enQueue(int element) {
     if (isFull()) {
       System.out.println("Queue is full");
     } else {
       if (front == -1)
         front = 0;
       rear++;
```

```
    items[rear] = element;
    System.out.println("Inserted " + element);
  }
}

int deQueue() {
  int element;
  if (isEmpty()) {
    System.out.println("Queue is empty");
    return (-1);
  } else {
    element = items[front];
    if (front >= rear) {
      front = -1;
      rear = -1;
    }
    else {
      front++;
    }
    System.out.println("Deleted element " + element);
    return (element);
  }
}

void display() {

  int i;
  if (isEmpty()) {
    System.out.println("Empty Queue");
  } else {
    System.out.println("Queue:");
    for (i = front; i <= rear; i++)
      System.out.print(items[i] + "  ");
  }
}

public static void main(String[] args) {
  trial q = new trial();
  System.out.println("22DCE006\n");
  q.enQueue(1);
  q.enQueue(2);
  q.enQueue(3);
  q.enQueue(4);
  q.enQueue(5);
  q.deQueue();
  q.display();
```

```
  }
 }
```

Output:

```
PS D:\Probin's Work\Java Programming> javac trial.java
PS D:\Probin's Work\Java Programming> java trial
22DCE006

Inserted 1
Inserted 2
Inserted 3
Inserted 4
Inserted 5
Deleted element 1
Queue:
2  3  4  5
PS D:\Probin's Work\Java Programming>
```

2) Queue Using LinkedList

Program Code:

```
public class college_dsa
{
 private Node front, rear;
 private int currentSize;
 private class Node
   {
      int data;
      Node next;
   }
 public college_dsa()
 {
 front = null;
 rear = null;
 currentSize = 0;
 }

 public boolean isEmpty()
 {
 return (currentSize == 0);
```

```
}

public void dequeue()
{
int data = front.data;

if (isEmpty())
{
rear = null;
}
 front = front.next;
currentSize--;
}

public void enqueue(int data)
{
Node oldRear = rear;
rear = new Node();
rear.data = data;
rear.next = null;
if (isEmpty())
{
front = rear;
}
else
{
oldRear.next = rear;
}
currentSize++;
}

public void display()
  {
     if(front==null)
     {
        System.out.println("Queue is empty");
        return;
     }
     Node temp=front;
     while (temp!= null)
     {
        System.out.println(temp.data+" ");
        temp=temp.next;
     }
  }
public static void main(String a[])
```

```
{

college_dsa queue = new college_dsa();
System.out.println("\n22DCE006\n");
queue.enqueue(1);
queue.enqueue(2);
queue.enqueue(3);
queue.enqueue(4);
queue.dequeue();
queue.display();
 }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac college_dsa.java
PS D:\Probin's Work\Java Programming> java college_dsa

22DCE006

2
3
4
PS D:\Probin's Work\Java Programming>
```

3)Circular Queue Using Array

Program Code:

```
public class q
{
   int size=5;
   int queue[]=new int[size];
   int front=-1;
   int rear=-1;

   public void enqueue(int data)
   {
      if(front==-1 && rear==-1)
      {
         front=rear=0;
         queue[rear]=data;
      }
      else if((rear+1)%size == front)
```

```
    {
      System.out.println("OVERFLOW ");
    }
    else
    {
       rear=(rear+1) % size;
       queue[rear]=data;
    }
}

public void dequeue()
{
    if(front==-1 && rear==-1)
    {
       System.out.println("UNDERFLOW");
    }
    else if( front==rear)
    {
       front=rear= -1;
    }
    else
    {
       front=(front+1) % size;
    }
}

public void display()
{
    System.out.println("QUEUE IS AS FOLLOWS: ");
    int i=front ;

    if(front==-1 && rear==-1)
    {
       System.out.println("QUEUE IS EMPTY");
    }
    else
    {
       while(i != rear)
       {
          System.out.print(" "+queue[i]);
          i=(i+1) % size;
       }
       System.out.print(" "+queue[rear]);
    }
    System.out.println();
}
```

```
    public static void main(String[] args) {
        q q1=new q();
        q1.enqueue(10);
        q1.display();
        q1.enqueue(20);
        q1.display();
        q1.enqueue(30);
        q1.display();
        q1.enqueue(40);
        q1.display();
        q1.enqueue(50);
        q1.display();
        q1.dequeue();
        q1.display();
        q1.dequeue();
        q1.display();
        q1.enqueue(1);
        q1.display();
        q1.enqueue(2);
        q1.display();
    }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> java q
QUEUE IS AS FOLLOWS:
 10
QUEUE IS AS FOLLOWS:
 10 20
QUEUE IS AS FOLLOWS:
 10 20 30
QUEUE IS AS FOLLOWS:
 10 20 30 40
QUEUE IS AS FOLLOWS:
 10 20 30 40 50
QUEUE IS AS FOLLOWS:
 20 30 40 50
QUEUE IS AS FOLLOWS:
 30 40 50
QUEUE IS AS FOLLOWS:
 30 40 50 1
QUEUE IS AS FOLLOWS:
 30 40 50 1 2
PS D:\Probin's Work\Java Programming> |
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-9

AIM: There are N people, numbered from 11 to N. They go to a cinema hall. Each of them buys a ticket, which has a number written on it. The number on the ticket of the ith person is Ai.
There are infinite seats in the cinema hall. The seats are numbered sequentially starting from 11. All the N people stand in a queue to get their respective seats. Person 11 stands at the front of the queue, Person 22 stands in the second position of the queue, so on up to Person N who stands at the rear of the queue. They were given seats in this manner: Let the number on the ticket of the person currently standing in front of the queue be X. If the X th seat is empty, the person gets out of the queue and takes the X th seat. Otherwise the person goes to the rear of the queue,
and the number on his ticket is incremented by one - that is, it becomes X+1.
Print the seat number occupied by each of the N people.
Input
• The first line of input contains a single integer T denoting the number of test cases. The description of T test cases follows.
• The first line of each test case contains a single integer N.
• The second line of each testcase line contains N space-separated integers A1, A2, ..., AN.
Output
• For each test case, print a single line containing N space-separated integers, where the I th integer denotes the seat number finally occupied by the Person i.
Input
4
5
1 2 3 2 4
4
4 1 3 2
3
1 1 1
5
2 5 1 5 2
Output
1 2 3 5 4
4 1 3 2
1 2 3
2 5 1 6 3

Program Code:

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
   int size;
   cout<<"22DCE006\n";
   cout<<"Enter the total no. of elements: ";
   cin >> size;
   vector<int> tickets(size);
   vector<int> seats(size);
   cout << "Value: ";
   for (int i = 0; i < size; i++)
   {
      cin >> tickets[i];
   }
   vector<bool> seatUsed(size + 1, false);
   for (int i = 0; i < size  ; i++)
   {
      int ticket = tickets[i];
      while (seatUsed[ticket])
      {
         ticket++;
      }
      seatUsed[ticket] = true;
      seats[i] = ticket;
   }
   for (int i = 0; i < size ; i++)
   {
      cout<<seats[i]<<" ";
   }
   return 0;
}
```

OUTPUT:

```
22DCE006
Enter the total no. of elements: 4
Value: 0
1
2
3
0 1 2 3
Process returned 0 (0x0)   execution time : 8.626 s
Press any key to continue.
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-10

Aim: Implement Binary Search Tree (BST) using following operations.
1) Insert
2) Search
3) Traversal(Inorder , Preorder , Postorder)

1) Program Code:

```
class tree {
 int data;
 tree left, right;

 tree() {
    data = 0;
    left = null;
    right = null;
 }

 tree(int value) {
    data = value;
    left = null;
    right = null;
 }

 tree Insert(tree root, int value) {
    if (root == null) {
       return new tree(value);
    }


    if (value > root.data) {

       root.right = Insert(root.right, value);
    } else if (value < root.data) {

       root.left = Insert(root.left, value);
    }
```

```
        return root;
    }

    void Inorder(tree root) {
        if (root == null) {
            return;
        }
        Inorder(root.left);
        System.out.print(" "+root.data);
        Inorder(root.right);
    }

    public static void main(String[] args) {
        System.out.println("22DCE006");
        tree b = new tree();
        tree root = null;
        root = b.Insert(root, 50);
        b.Insert(root, 30);
        b.Insert(root, 20);
        b.Insert(root, 40);
        b.Insert(root, 70);
        b.Insert(root, 60);
        b.Insert(root, 80);

        b.Inorder(root);

    }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> javac tree.java
PS D:\Probin's Work\Java Programming> java tree
22DCE006
 20 30 40 50 60 70 80
PS D:\Probin's Work\Java Programming>
```

2)Program Code:

```
public class tree {
    public int key;
    public tree lft, rgt;

    public tree() {
        key = 0;
        lft = null;
        rgt = null;
    }

    public tree(int value) {
        key = value;
        lft = rgt = null;
    }

    public tree insertFunc(tree root, int value) {
        if (root == null) {
            return new tree(value);
        }
        if (value > root.key) {
            root.rgt = insertFunc(root.rgt, value);
        } else {
            root.lft = insertFunc(root.lft, value);
        }
        return root;
    }

    public tree searchFunc(tree root, int key) {
        if (root == null || root.key == key) {
            return root;
        }
        if (root.key < key) {
            return searchFunc(root.rgt, key);
        }
        return searchFunc(root.lft, key);
    }

    public void traverseInOrder(tree root) {
```

```
    if (root == null) {
        return;
    }
    traverseInOrder(root.lft);
    System.out.println(root.key);
    traverseInOrder(root.rgt);
}

public static void main(String[] args) {
    System.out.println("22DCE006");
    tree node = new tree();
    tree root = null;
    tree searchRoot = null;

    root = node.insertFunc(root, 13);
    node.insertFunc(root, 15);
    node.insertFunc(root, 17);
    node.insertFunc(root, 19);


    System.out.println("\nThe sorted binary search tree is:");
    node.traverseInOrder(root);

    System.out.println("\nSearch for 17 in the BST:");
    searchRoot = node.searchFunc(root, 17);
    if (searchRoot == null) {
        System.out.println("Value Not Found");
    } else {
        System.out.println("Value Found! " + searchRoot.key);
    }

    System.out.println("\nSearch for 19 in the BST:");
    searchRoot = null;
    searchRoot = node.searchFunc(root, 19);
    if (searchRoot == null) {
        System.out.println("Value Not Found");
    } else {
        System.out.println("Value Found! " + searchRoot.key);
    }
```

```
    }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> javac tree.java
PS D:\Probin's Work\Java Programming> java tree
22DCE006

The sorted binary search tree is:
13
15
17
19

Search for 17 in the BST:
Value Found! 17

Search for 19 in the BST:
Value Found! 19
PS D:\Probin's Work\Java Programming>
```

3)Program Code:

```
class Node {
  int item;
  Node left, right;

  public Node(int key) {
  item = key;
  left = right = null;
  }
}

public class tree
{
  Node root;
  tree() {
  root = null;
  }
  void postorder(Node node) {
```

```
if (node == null)
  return;
postorder(node.left);
postorder(node.right);
System.out.print(" -- "+node.item );
}

void inorder(Node node) {
if (node == null)
  return;
inorder(node.left);
System.out.print(" -- "+node.item );
inorder(node.right);
}

void preorder(Node node) {
if (node == null)
  return;
System.out.print(" -- "+node.item );
preorder(node.left);
preorder(node.right);
}

public static void main(String[] args)
{
  System.out.println("22DCE006");
  tree tree = new tree();
  tree.root = new Node(1);
  tree.root.left = new Node(2);
  tree.root.right = new Node(3);
  tree.root.left.left = new Node(4);
  tree.root.left.right = new Node(5);

  System.out.println("Inorder traversal");
  tree.inorder(tree.root);

  System.out.println("\nPreorder traversal ");
  tree.preorder(tree.root);
```

```
    System.out.println("\nPostorder traversal");
    tree.postorder(tree.root);
  }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> javac tree.java
PS D:\Probin's Work\Java Programming> java tree
22DCE006
Inorder traversal
 -- 4 -- 2 -- 5 -- 1 -- 3
Preorder traversal
 -- 1 -- 2 -- 4 -- 5 -- 3
Postorder traversal
 -- 4 -- 5 -- 2 -- 3 -- 1
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-11

Aim: Implement a Graph to perform following operations.
1. Adjacency list representation
2. Apply DFS and BFS on the given graph.

1) Program Code:

```
import java.util.*;
class graph
{
   public static void addEdge(List<Integer>[] adj, int u, int v)
   {
      adj[u].add(v);
      adj[v].add(u);
   }


   public static void printGraph(List<Integer>[] adj, int V)
   {
      for (int v = 0; v < V; ++v)
      {
         System.out.println("\n Adjacency list of vertex " + v + "\n head ");
         for (int x : adj[v])
           {  System.out.print("-> " + x);}
         System.out.println();
      }
   }

   public static void main(String[] args)
   {
       System.out.println("22DCE006");
      int V = 5;
      List<Integer>[] adj = new ArrayList[V];
      for (int i = 0; i < V; i++)
      {
         adj[i] = new ArrayList<>();
      }
      addEdge(adj, 0, 1);
```

```
        addEdge(adj, 0, 2);
        addEdge(adj, 1, 2);
        addEdge(adj, 1, 3);
        addEdge(adj, 1, 4);
        addEdge(adj, 2, 3);
        addEdge(adj, 3, 4);
        printGraph(adj, V);

    }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> java graph
22DCE006

 Adjacency list of vertex 0
 head
-> 1-> 2

 Adjacency list of vertex 1
 head
-> 0-> 2-> 3-> 4

 Adjacency list of vertex 2
 head
-> 0-> 1-> 3

 Adjacency list of vertex 3
 head
-> 1-> 2-> 4

 Adjacency list of vertex 4
 head
-> 1-> 3
```

2) M-1)BFS

Program Code:

```
import java.util.*;

public class graph {
  private int V;
  private LinkedList<Integer> adj[];
  graph(int v) {
    V = v;
    adj = new LinkedList[v];
    for (int i = 0; i < v; ++i)
      adj[i] = new LinkedList();
  }
  void addEdge(int v, int w) {
    adj[v].add(w);
  }
  void BFS(int s) {

    boolean visited[] = new boolean[V];

    LinkedList<Integer> queue = new LinkedList();

    visited[s] = true;
    queue.add(s);

    while (queue.size() != 0) {
      s = queue.poll();
      System.out.print(s + " ");

      Iterator<Integer> i = adj[s].listIterator();
      while (i.hasNext()) {
        int n = i.next();
        if (!visited[n]) {
          visited[n] = true;
          queue.add(n);
        }
      }
```

```
    }
  }
  public static void main(String args[]) {
    System.out.println("22DCE006");
    graph g = new graph(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);
    System.out.println("Following is Breadth First Traversal " + "(starting from vertex
2)");
    g.BFS(2);
  }
}
```

OUTPUT:

```
3 warnings
PS D:\Probin's Work\Java Programming> java graph
22DCE006
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
PS D:\Probin's Work\Java Programming>
```

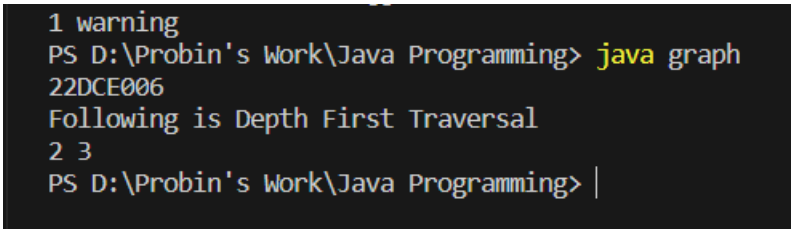
M-1)DFS

Program Code:

```
import java.util.*;
class graph {
  private LinkedList<Integer> adjLists[];
  private boolean visited[];
  graph(int vertices) {
    adjLists = new LinkedList[vertices];
    visited = new boolean[vertices];

    for (int i = 0; i < vertices; i++){
      adjLists[i] = new LinkedList<Integer>();
  }
```

```
 void addEdge(int src, int dest)
 {
   adjLists[src].add(dest);
 }
 void DFS(int vertex)
{
   visited[vertex] = true;
   System.out.print(vertex + " ");

   Iterator<Integer> ite = adjLists[vertex].listIterator();
   while (ite.hasNext())
   {
     int adj = ite.next();
     if (!visited[adj])
       DFS(adj);
   }
 }

 public static void main(String args[])
 {
   System.out.println("22DCE006");
   graph g = new graph(4);
   g.addEdge(0, 1);
   g.addEdge(0, 2);
   g.addEdge(1, 2);
   g.addEdge(2, 3);
   System.out.println("Following is Depth First Traversal");
   g.DFS(2);
 }
}
```

OUTPUT:



Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-12

Aim: In an array of 20 elements, arrange 15 different values, which are generated randomly between 1,00,000 to 9,99,999. Use hash function to generate key and linear probing to avoid collision. H(x) = (x mod 18) +2. Write a program to input and display the final values of array.

Program Code:

```
import java.util.*;
public class h
{
   public static int hashFunction(int x)
   {
      return (x % 18) + 2;
   }

   public static void main(String[] args)
   {
      System.out.println("22DCE006");
      final int SIZE = 20;
      int[] arr = new int[SIZE];
      int num, key, probes;
      Random r1 = new Random();
      for (int i = 0; i < 15; i++) {
         num = r1.nextInt(900000) + 100000;
         key = hashFunction(num);
         probes = 0;

         while (arr[key] != 0 && probes < SIZE)
         {
            key = (key + 1) % SIZE;
            probes++;
```

```
            }
            if (probes == SIZE) {
                System.out.println("Error: Array is full");
                System.exit(1);
            }
            arr[key] = num;
        }

        System.out.println("Final values of the array:");
        for (int i = 0; i < SIZE; i++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}
```

OUTPUT:

```
PS D:\Probin's Work\Java Programming> javac h.java
PS D:\Probin's Work\Java Programming> java h
22DCE006
Final values of the array:
306320 893373 0 0 0 154425 955048 185025 931182 456541 586771 608859 832149 490653 0 0 356018 264345 594502 587894
PS D:\Probin's Work\Java Programming>
```

Staff Signature:

Grade:

Remarks by the Staff: